Программирование

Python

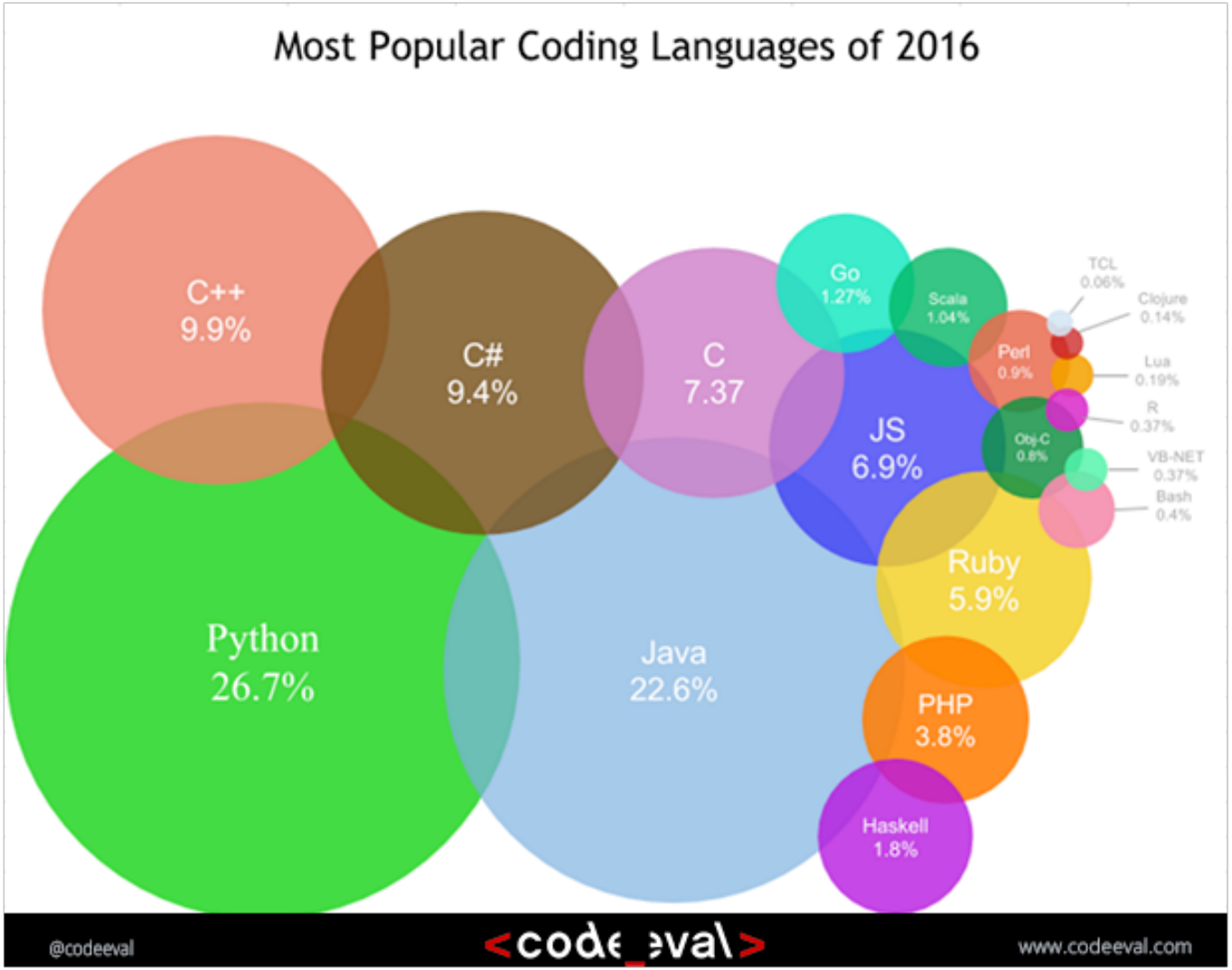
Кафедра ИВТ и ПМ

2017

План

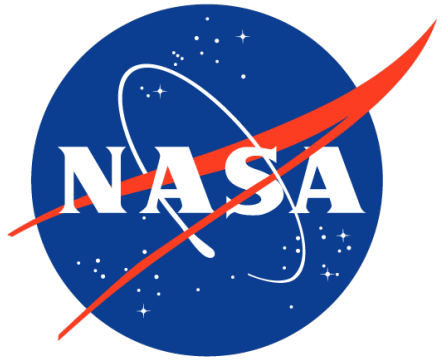
- О Python
- Интерактивный режим
- Введение в язык
  - Математические функции
  - Литералы
  - Переменные
  - Операции
  - Определение типа
- IDE - интегрированные среды разработки
- Основные операторы
  - Особенности
  - Условный оператор
  - Циклы

Most Popular Coding Languages of 2016



source

Кто использует?



Organizations Using Python

О Python

- Высокоуровневый язык
- Общего назначения
- Ориентирован на повышение производительности разработчика

- Интерпретируемый
- Компилируется в байт-код

- Объектно ориентированное программирование
- Функциональное программирование
- Рефлексивность
- Динамическая типизация
- Сборка мусора

Философия

import this

- Простое лучше, чем сложное.
- Сложное лучше, чем запутанное.
- Плоское лучше, чем вложенное.
- Разреженное лучше, чем плотное.
- Читаемость имеет значение.
- Встретив двусмысленность, отбрось искушение угадать.
- Должен существовать один — и, желательно, только один — очевидный способ сделать это.
- Сейчас лучше, чем никогда.
- Если реализацию сложно объяснить — идея плоха.
- Если реализацию легко объяснить — идея, возможно, хороша.
- ...

Как работать с Python?

- В интерактивном режиме
  Каждая команда (может быть составной) выполняется тут же

- Программы в отдельном файле

Распространены две версии языка: 3.x и 2.x.
Версия 2.x до сих порт используется только из-за того, что на ней было написано много кода на момент появления 3-й версии, которая внесла существенные изменения в язык.

Интерактивный режим

Использование python как калькулятора в интерактивном режиме

- работает как командная строка
- умеет производить математические вычисления
- результат вычисления сразу отображается на экране
- результат вычисления автоматически (неявно) записывается в переменную _ (знак подчёркивания)
- чтобы просмотреть содержимое переменной достаточно указать её имя (идентификатор)

```
>>> имяпеременной
```

Интерактивный режим

Использование python в качестве калькулятора

- Доступные арифметические операции: +, -, *, /
- Возведение в степень, x^y

  `x**y`

- Целочисленное деление

  ```
  7 // 3
  ```

  $= 2$

- Остаток от деления

  ```
  12 % 5
  ```

Outline

О Python

Интерактивный режим

Введение в язык

- Математические функции
- Литералы
- Переменные
- Операции
- Определение типа

IDE - интегрированные среды разработки

Основные операторы

- Особенности
- Условный оператор
- Циклы

Интерактивный режим

чтобы использовать математические функции и константы нужно подключить пакет math

```
import math
```

Чтобы вызывать функцию из этого модуля следует указать его имя и уже потом, через точку, имя функции
Например:

```
import math
math.sin ( math.pi / 2 )  # 1.0
math.degrees( pi / 2 )  # 90. радианы -> градусы.
math.radians( 90 )  # 3.14. градусы -> радианы
math.exp ( 2 )  # e^2 = 7.39
math.sqrt(81)  # 9
```

решётка - знак комментария.
Пакет в python - это набор модулей объединённых под одним именем.

Интерактивный режим

Однако каждый раз указывать имя пакета недобно. Поэтому можно при его подключении указать конкретные функции, что будут использованы:

```
from math import tan, sin
```

Если функций много, то их перечислять необязательно. Вместо этого можно сделать доступным всё содержимое пакета:

```
from math import *
```

Теперь можно использовать функции непосредственно, без указания имени пакета:

Интерактивный режим

Демонстрация

Типы

В Pyhton не так много встроенных типов

- bool
- int
- float

Составные типы

- str - строка
- tuple - кортеж
- list - список
- map - отображение (ассоциативный массив)

Outline

- О Python
- Интерактивный режим
- Введение в язык
  - Математические функции
  - Литералы
  - Переменные
  - Операции
  - Определение типа
- IDE - интегрированные среды разработки
- Основные операторы
  - Особенности
  - Условный оператор
  - Циклы

Литералы

- Целочисленные литералы

```
123
0
9999999999999999999  # точность не ограничена
0o177  # вшсьмиричное число (Oct)
0x9ff  # Шестнадцатеричное число (heX)
0b101010  # двоичное число (Bin)
```

Литералы

- Литералы вещественного типа

```
1.23
1.
.123
3.14e-10   # 3.14 ^-10
4E210      # 4^210
4.0e+210   # 4^210
```

Литералы

- Литералы комплексного типа

```
3 + 4j       # 3 + 4i
3.0 + 4.0j   # то же самое

# можно писать большую J
3J           # 0 + 3i.
2 + 0j       #
```

Для обозначения мнимой части комплексного числа используют постфикс **j**.

Литералы

- Литералы логического типа

  `True`
  `False`

  Регистр имеет значение.

Литералы

- Строковые литералы

```
'qwerty'   # можно использовать одинарные кавычки
"abcdef"   # а можно двойные

# в двойных кавычках одинарные считаются отдельным с
"Can't"

# и наоборот
'ФГБОУ ВО "ЗабГУ"'

"""текст в тройных кавычках может располагаться
на нескольких строках.
ещё одна строка
и ещё"""
```

Неправильно:

```
'кавычки должны соответствовать друг другу"
```

Outline

- О Python
- Интерактивный режим
- Введение в язык
  - Математические функции
  - Литералы
  - Переменные
  - Операции
  - Определение типа
- IDE - интегрированные среды разработки
- Основные операторы
  - Особенности
  - Условный оператор
  - Циклы

Переменные

В Python для объявления переменной достаточно задать её значение.
Регистр имеет значение: A и a - разные переменные.
Оператор присваивания =

```
a = 42     # переменная целого типа
x = 3.14   # ... вещественного типа
s = "Hello, Python!"  # ... строкового типа
b = True   # ... логического типа
```

Тип переменной не указывается. Python сам выбирает тип, в зависимости от заданного значения.

Переменные

Определить тип переменной (см. предыдущий слайд)

```
type(a)   # int
type(x)   # float
type(s)   # str
type(b)   # bool
```

Переменные

Нельзя объявить переменную не задав ей значения
Если переменную объявить нужно, но пока не ясно какое значение ей задать можно использовать пустое значение None:

```
a = None
```

Outline

- О Python
- Интерактивный режим
- Введение в язык
  - Математические функции
  - Литералы
  - Переменные
  - Операции
  - Определение типа
- IDE - интегрированные среды разработки
- Основные операторы
  - Особенности
  - Условный оператор
  - Циклы

Операции с вещественным типом

- Арифметические
  +, -, *, // (целочисленное деление), % (остаток от деления), **
- Сравнение
  == (равно), != (не равно), <, >, <=, >=
- Принадлежность диапазону

  ```
  x < y < z
  ```

  то же самое что и

  ```
  x < y and y < z
  ```

Операции с целым типом

- те же самые, что и с вещественным типом
- побитовые операции
  (возведение в степень) » (побитовый сдвиг вправо), «
  (побитовый сдвиг влево)

Outline

- О Python
- Интерактивный режим
- Введение в язык
  - Математические функции
  - Литералы
  - Переменные
  - Операции
  - Определение типа
- IDE - интегрированные среды разработки
- Основные операторы
  - Особенности
  - Условный оператор
  - Циклы

Типы и преобразование типов

Какие типы будут у этих переменных?

```
x = 300 + 2.0
y = 12 + 8 // 2
z = sin( pi ) + 1
a = 12.0 // 4
b = 12.0 % 5
```

Преобразование типов

Какие типы будут у этих переменных?

```
x = 300 + 2.0        # float
y = 12 + 8 // 2      # int
z = sin( pi ) + 1    # float
a = 12.0 // 4        # float
b = 12.0 % 5         # float
```

Преобразование типов

Тип определяется по типу литерала.
Тип всегда приводится к наиболее общему.
Тип float является более общим чем int.
Тип complex является более общим для float.
Например при сложение вещественного и целого числа результат будет вещественного типа

Явное преобразование типов

`int(выражение)`

преобразует выражение в целое число

```
int("123")   # строка -> целое, 123
int(5.125)   # вещественное число -> целое.
# дробная часть отбрасывается

float("123") # строка -> вещесвтв. число, 123.0
float(123)   # 123 -> 123.0

str(123)   # 123 -> "123"
str(5.125)   # 5.125 -> "5.125"
```

IDE - интегрированные среды разработки

- Wing
- Jupiter (Anaconda)
- PyCharm
- ...

Ввод и вывод

Вывод

```
print( ... )
```

```
print ( "Hello, Python!" )
```

```
    a = 10
    b = 3.14
    print("a = ", a, "; b = ", b)
    # будет напечатано:
    # a = 10; b = 3.14
```

Вывести на экран и не переходить на следующую строку:

```
print ( "Hello, Python!", end="")
```

pythoner.name/formatted-output - форматирование вывода

Ввод и вывод

Ввод данных

```
s = input()
```

Функция input ожидает ввод с клавиатуры и возвращает строковое значение. Чтобы преобразовать строку в целое и

вещественное число нужно использовать явное преобразование типа:

```
d = int( input() )
```

```
f = float( input() )
```

Outline

- О Python
- Интерактивный режим
- Введение в язык
  - Математические функции
  - Литералы
  - Переменные
  - Операции
  - Определение типа
- IDE - интегрированные среды разработки
- Основные операторы
  - Особенности
  - Условный оператор
  - Циклы

Особенности синтаксиса языка

После инструкций точка с запятой не ставится.
Одна строка - одна инструкция
Конец строки - конец инструкции
Пример последовательности инструкций (операторов):

```
a = 42
b = 13
c = a + b

print("a + b = ", c)
```

Пустые строки игнорируются

Особенности синтаксиса языка

Отступы - это часть синтаксиса языка
Операторных скобок нет
Уровни вложенности определяются отступами. Для одного уровня вложенности рекомендуется использовать отступ из 4-х пробелов.

```
if a > b:
    print("a > b")

while a > b:
    a = a + 1
```

Перед вложенными операторами всегда ставится двоеточие.

Outline

- О Python
- Интерактивный режим
- Введение в язык
  - Математические функции
  - Литералы
  - Переменные
  - Операции
  - Определение типа
- IDE - интегрированные среды разработки
- Основные операторы
  - Особенности
  - Условный оператор
  - Циклы

Условный оператор

Сокращённая форма

```
if лог.выражение :
    оператор1
```

Условный оператор

Полная форма

```
if лог.выражение :
    оператор1
else:
    оператор2
```

Условный оператор

Операторов может быть несколько.
Все они должны выделятся отступами

```
if лог.выражение:
    оператор1
    оператор2
    оператор3

оператор4
```

оператор4 будет выполнен в любом случае.

Логические выражения

Операции с логическим типом
and, or, not, ^(xor)

```
x = 5
y = 6
z = 5

x < y and x == z   # True
x != y   # True
x == 6 or x == 7   # True
x < y < z   # False

True ^ False   # True
```

Условный оператор

Определить максимальное из двух *различных* чисел a и b.

```
max = None
if a > b:
    max = a
else:
    max = b

print( "max = ", max )
```

Короткий вариант

Условный оператор

Определить максимальное из двух *различных* чисел a и b.

```
max = None
if a > b:
    max = a
else:
    max = b

print( "max = ", max )
```

Короткий вариант

```
max = a
if b > max:
    max = b


print( "max = ", max )
```

Outline

- О Python
- Интерактивный режим
- Введение в язык
  - Математические функции
  - Литералы
  - Переменные
  - Операции
  - Определение типа
- IDE - интегрированные среды разработки
- Основные операторы
  - Особенности
  - Условный оператор
  - Циклы

Циклы

- Цикл с предусловием. while
- Цикл со счётчиком (совместный цикл). for

Циклы

Цикл с предусловием. Короткая форма

```
while условие:
    оператор1
```

Оператор1 будет выполнятся до тех пор пока условие истинно.

Циклы

Цикл с предусловием. Полная форма

```
while условие :
    оператор1
else:
    оператор2
```

Оператор2 выполнится когда условие станет ложным.

Циклы

Цикл for
Этот цикл может выполнять обход элементов последовательностей.
Например перебрать все элементы в массиве или символы в строке.

```
for e in последовательность :
    оператор1
```

Оператор1 будет выполнятся пока не будут перебраны все элементы в последовательности.
Переменную цикла e объявлять заранее не нужно.

Циклы

Создание простых последовательностей

range(n) - создаёт последовательность из n целых чисел 0..n-1
range(a, b) - создаёт последовательность из b-a целых чисел от

a..b-1
Последние число никогда не включается в последовательность.
b должно быть больше чем **a**

```
range(5)      # 0, 1, 2, 3, 4
range(0, 5)   # 0, 1, 2, 3, 4

range(3,7)    # 3, 4, 5, 6
range(-3, 1) # -3, -2, -1, 0
```

Циклы

Напечатать цифры от 0 до 10

```
for i in range(11):
    print(i)
```

Списки

В Python нет встроенного типа *массив*.
Вместо них - списки.
Синтаксис обращения к элементам списка похож на синтаксис массивов.
Объявить пустой массив:

```
l = []
```

Можно сразу задать значения:

```
l = [ 1, 2, 3 ]
```

Списку не нужно задавать начальный размер потому, что в любой момент можно добавить ещё один элемент в конец.

```
l = l + [42] ## l = [1, 2, 3, 42]
```

Списки

Некоторые операции со списками

```
l = [10, 20, 30, 42]

n = len(l)  # получить длинну списка
x = l[2]  # доступ к элементам. индексация с 0
# x = 30

z = l[-1]  # доступ к последнему элементу

l2 = [0] * 128  # создание списка из 128 нулей
```

Списки

Создать список из n случайных чисел

```
from random import random
# функция random возвращает псевдослучайное число
# в интервале от 0 до 1
```

Традиционный способ

```
n = 100
l = []
for i in range(n):
    l = l + [random()]
```

Списки

Быстрее и короче. С помощью генератора списка.

```
n = 100
l = [random() for i in range(n)]
```

Списки

Проход по списку.
"Традиционный способ"

```
l = [1,2,3,4]

for i in len(l):
    print(l[i])
```

С помошью совместного цикла

```
l = [1,2,3,4]

for e in l:
    print(e)  # в теле такого цикла нельзя менять e
```

Двумерный список

```
m = [ [1,2,3], [4,5,6], [7,8,9] ]
for l in m:
    for e in l:
        print(e, end="")
    print()
```

Ссылки и литература

ЭБС

- biblio-online.ru - ЭБС Юрайт
- studentlibrary.ru - ЭБС "КОНСУЛЬТАНТ СТУДЕНТА"

- Федоров, Д. Ю. Программирование на языке высокого уровня python : учебное пособие для прикладного бакалавриата Содержит краткое описание языка.
- ru.wikibooks.org/wiki/Python - Викиучебник
- Лутц М. Изучаем Python. 2010. - 1280 с. Содержит подробное описание языка.
- Официальная документация Python3

  `help( имя )`

Дополнительная литература

- O'Conor T.J. Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers. 2012 — 288 p.
- Numerical methods in engineering with Python 3 / Jaan Kiusalaas.